Mobile AR in the outdoors: watch the clouds

Pieter Simoens[*,†], Tim Verbelen[†], Bart Dhoedt[†]
[*] Ghent University College, Dept. INWE
[†] Ghent University - IBBT, Dept. of Information Technology

Abstract

Even the most modern wearable devices do not provide sufficient resources for the complexity of outdoor AR algorithms, induced by the uncontrolled environment. In this paper, we present a framework whereby nearby infrastructure, called cloudlets, is leveraged with an execution environment to provide additional computational power and storage capacity. The execution environment is dynamically composed with service components and data, dependent on the actual user context. We discuss the architecture and provide first experimental results on a distributed mobile AR application that has been adapted to the cloudlet paradigm.

1. Introduction

Offering a plethora of rich sensing capabilities in an easily wearable device, smartphones are a highly attractive platform for mobile augmented reality applications. With quad-core processors and gigabytes of memory, modern smartphones are computationally capable to perform real-time detection and tracking of objects in AR applications. Limitations in size, weight and heat dissipation make mobile devices intrinsically resource poor, compared with desktop or server infrastructure [8]. For example, using the GPU on a smartphone rapidly drains the battery and prohibitively reduces the autonomity - and thus mobility - of its users. As a result, mobile AR developers need to take measurements that hamper the user experience, e.g. by sacrificing the accuracy and quality of state-of-the-art algorithms to meet latency requirements [3], or by introducing an offline preprocessing stage on a remote server [1].

In outdoor environments, the computational and data requirements of mobile AR applications are even more challenging. Compared with indoor environments, variable and difficult lighting conditions require particular detection algorithms, objects are typical of larger scale, e.g. 3D construction information of buildings [2], and the larger area explored by users contains a multiple of the number of objects to be recognized. Besides computational and storage constraints, outdoor mobile AR is typically based on time-varying user context environment, as demonstrated by Google's Project Glass [6] whereby the application continuously reacts to new events in the user's vicinity.

To overcome smartphone resource limitations and to cope with the larger scale of complexity in outdoor environments, mobile AR applications need to be supported by the cloud. Already in the early days of mobile AR development, researchers acknowledged the need to offload computationally intensive parts to a nearby server that was typically accessed through a local wireless access point. With the advent of cloud computing and broadband 3G/4G wireless connections, also in outdoor environments external resources are within reach. Furthermore, levering AR applications with cloud-based components allows to analyze in real-time the current context and provide the required updates upon unanticipated events to the mobile device. Engaging the cloud introduces however a number of challenges. The latency between mobile device and cloud may be prohibitive for applications with strict real-time constraints. Also, a lot of information is only relevant for a limited period in time or for a small number of users, e.g. dependent on the geographical distance.

In this paper, we outline our vision on how the cloud can support mobile AR. We are working on a management framework that deploys for each user an execution environment on nearby infrastructure, called cloudlets. The execution environment is dynamically composed with application components and data, based on context parameters such as geographical location, time or available wireless bandwidth. Time-critical application components and personal data are optimally distributed over the mobile device and the execution environment on the cloud, while background tasks and large object database are centrally managed in the cloud. The proposed framework is an extension of the cloudlet concept coined by Satyanarayanan et al. [8]. The authors use the virtual machine as unit of deployment and use a thin client approach. In contrast, we exploit the increased computational capabilities of smartphones to run the most time-critical application components on the device. Furthermore, offloading at component level offers

more flexibility to cope with the dynamic context parameters and heterogeneous devices in mobile environments.

The remainder of this paper is structured as follows. In section 2, we describe the architecture of our framework and the component-based approach. In section 3 we present our current implementation. In section 4, we present experimental results on the performance of a distributed mobile AR application, clearly demonstrating the benefits of the cloudlet approach. In section 5, we repeat the most relevant conclusions and outline future work.

2. Proximate cloud model

Figure 1 illustrates our vision in which nearby resources are leveraged with an execution environment to provide the required computational and storage for latency-critical tasks. Instead of running the complete application on the mobile device, application components can be offloaded to the cloudlet or the cloud. Different cloudlet configurations are supported. Some cloudlets are formed by already installed devices with network connectivity that have underutilized resources. In a home network, many devices are left always on, e.g. for home network media file sharing, but rarely require all available resources. Given the widespread deployment of Wi-Fi in residential areas, there are few places where a mobile device is out of reach of any of these underutilized devices. Collections of mobile devices may form an ad-hoc cloudlet, e.g. in a train carriage where a lot of mobile devices remain in the same configuration for a relatively long period. Of course, sharing resources of personal devices introduces challenges of trust and energy consumption. We believe these challenges can be overcome with virtualization techniques becoming available on mobile devices [7], and the establishment of an incentive-based mechanism following the tit-for-tat principle as is found in P2P networks [9]. Other cloudlets may be composed with dedicated infrastructure, e.g. current public Wi-Fi hotspots could be enhanced with a server providing resources, or mobile operators may integrate infrastructure in their mobile aggregation network.

As the user moves, the execution environment on the cloudlet may contain different application components and data. Operating at the component level, this architectural framework provides the required flexibility for outdoor mobile AR applications with continuous context changes. When the user moves, the execution environment can be dynamically reconfigured. This allows for example to load the execution environment with location-specific service components and data, e.g. an augmented reality tourist guide. Moreover, information and services can be shared between multiple users at the same location. For example, in tracking applications, users could reuse the map that has been previously constructed by others at the same location.

3. Implementation

We implemented a prototype of our framework in Java, which allows it to run on a wide range of platforms, including Android, the popular mobile operating system. We built the execution environment with OSGi, a service oriented module management system that allows to dynamically register software modules. OSGi offers life cycle component management, automatic resolution of dependencies and, using R-OSGi, the distribution of components across different OSGi instances.

Each application component is registered as an OSGi bundle to the execution environment. For each bundle, the providing interfaces are proxied by the Execution Environment (EE). Components receive a reference to the proxy, which allows the EE to transparently monitor all calls to the proxied object, and to choose whether to forward incoming method calls to the local object or to a remote duplicate. Consequently, all application components can be transparently distributed between the cloudlet and the mobile device.

For each component, the application developer must provide an XML description specifying performance constraints, such as maximum processing time per frame, and the range of configurable parameters, e.g. the number of feature points to analyze. At runtime, the performance of the application may vary because of many factors, e.g. wireless bandwidth fluctuations, more complex scenes to analyze, or periodic background tasks on the mobile device. The framework will react by reconfiguring parameters or by recalculating the optimal deployment of components between EEs deployed on the mobile device and the cloudlet. In the current implementation only predefined actions are configured. More details on decision algorithms for distributing components with multiple configuration options can be found in our previous work [10].

4. Use case: distributed PTAM

To illustrate the merits of our 3-tier approach, we have combined an existing augmented reality application using markerless tracking (based on the Parallel Tracking And Mapping (PTAM) algorithm [5]), with an object recognition algorithm [4]. PTAM is a widely used tracking algorithm in a priori unknown environments and is hence a good candidate to be incorporated in outdoor mobile AR applications. A screenshot of the application is shown in Figure 2. The application continuously analyzes frames captured by the camera of the mobile device to estimate its current pose. Recognized objects are indicated with a white border and a 3D object is rendered as an overlay.

The application comprises the following components:

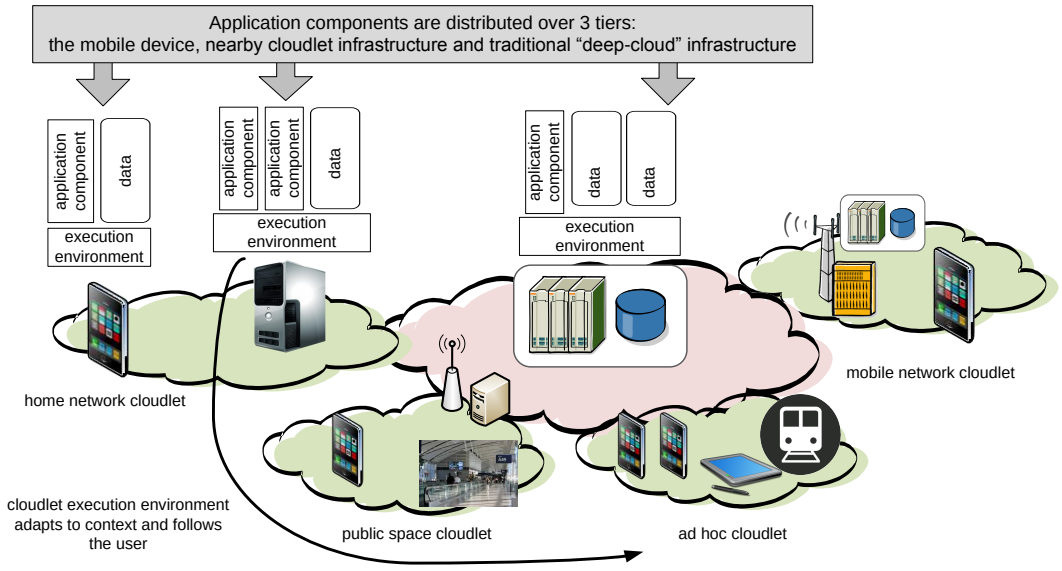

Figure 1. Application components and data are distributed over the mobile device, the cloudlet and the cloud. This allows to differentiate between latency critical components and background tasks.

Figure 2. The application tracks feature points (right) to enable the overlay of 3D objects (left).

- **VideoSource** fetches video frames from the camera hardware. These frames are forwarded to the Tracker for analysis, and rendered with an augmented reality overlay by the Renderer.
- **Renderer** outputs the captured frames and the generated 3D overlay to the camera display. The 3D objects are aligned according to the camera pose that is estimated by the Tracker.
- **Tracker** calculates the camera pose by matching a set of 2D image features to a known map of 3D feature points, maintained by the Mapper.
- **Mapper** At regular intervals, the Tracker sends frames to the Mapper for map generation and refinement. By matching 2D features in a set of keyframes, the Mapper estimate their 3D location in the scene and generates a map of 3D feature points.
- **Relocalizer** tries to relocate the camera position when no feature points are found in the captured frames until the tracking can be resumed
- **Object Recognizer** tries to locate known objects in the keyframes of the Mapper. The 3D location of recognized objects is notified to the Renderer that produces an appropriate white boundary around the recognized object.

Our current implementation exists of a smartphone and a laptop taking the role of cloudlet server. The laptop is equiped with an Intel Core 2 Duo 2.26 GHz running Linux. The smartphone is an Android LG Optimus 2x with a dual core Nvidia Tegra 2 CPU of 1 GHz. The different components have been implemented as OSGi bundles. To speedup the computation, an important part of the image processing is implemented in native C/C++ code that was wrapped in OSGi components by means of the Java Native Interface. Native libraries have been compiled for both the x86 and ARM processor architectures. The appropriate library depends on the underlying platform and is loaded at runtime by the OSGi framework.

We evaluated 3 deployment configurations of the AR components. In configuration A, all components are running on the mobile device. In the configuration B, the components not having real-time constraints, i.e. the Mapper

Table 1. Execution time in ms in 3 deployment scenarios. The resolution of the captured frames is 800x480.

configuration	A	B	C
Tracker	64	34	378
Relocalizer	21	21	2
Map init	2 804	1 124	176
Mapper	2 470	519	396
Object Recognizer	18 682	1 495	2 281

and the ObjectRecognizer, are outsourced to the laptop. In configuration C, also the Tracker and Relocalizer are offloaded to the laptop, leaving only the components that access device specific hardware on the device.

Table 1 presents the execution time of the individual components for each configuration. The results represent the duration of a single operation of the component: tracking one frame (Tracker), relocalizing one frame (Relocalizer), initializing the 3D map (Map Init), refining the map with one keyframe (Mapper) and for object detection in one frame (Object Recognizer).

The duration to track a single frame is the most important factor affecting the user experience, since this has to be performed for each camera frame that is fetched. Only in deployment configuration B, an acceptable framerate of 30 fps is realized. Compared with scenario A, outsourcing the Mapper and ObjectRecognizer to the laptop allows to allocate more resources to the Tracker. On the other hand, outsourcing the Tracker component to the laptop drastically increases the execution time of the Tracker since in this case, the captured frames as well as the results must be transferred over the wireless connection. The same observation holds for the Object Recognizer, which receives its frames from the Tracker. In scenario C, both components are colocated, whereas in scenario B these frames have to be transmitted over the wireless network. These results clearly indicate the advantage of our fine-grained component based approach, allowing much more flexibility in deployment configurations. Overall, outsourcing components results in a reduction of the execution time with a factor 2 to 12.

5. Conclusion and future work

Despite rapid improvements of mobile device hardware capabilities, the stringent computational and data requirements of mobile outdoor AR applications can only be provided by the cloud. In this paper, we have presented our framework in which users can outsource components to an execution environment on nearby cloudlet infrastructure. The offloaded application components and data may vary depending on the user context. Many research questions are currently being investigated. A first set of challenges relates to the management of the cloudlet and its composing nodes. As the user moves, his mobile device needs to connect with other cloudlets. One solution is to automatically select the most powerful node in the cloudlet (e.g. the desktop in a home network) as master node managing the cloudlet. Furthermore, we are working on algorithms that determine the optimal balancing of application components between the mobile device, the cloudlet and the cloud, based on runtime parameters as battery capacity, network bandwidth and application load. A last set of research questions is related to the reuse of service components and data between multiple users. Specifically for mobile AR, we are implementing the use case whereby mapping information is shared between multiple users to enable collaborative outdoor AR.

References

[1] C. Arth et al. Real-Time Self-Localization from Panoramic Images on Mobile Devices. In *10th International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011.

[2] C. Woodward et al. Mobile augmented reality for building and construction. In *VTT Technical Research Centre of Finland. Mobile AR Summit@ MWC*, 2010.

[3] D. Wagner et al. Real-Time Detection and Tracking for Augmented Reality on Mobile Phones. *IEEE Trans. on Visualization and Computer Graphics*, 16(3), 2010.

[4] D.G. Lowe et al. Distinctive image features from scale-invariant keypoints. *Intl. journal of computer vision*, 60(2), 2004.

[5] G. Klein et al. Parallel tracking and mapping for small ar workspaces. In *6th International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007.

[6] Google. Project glass, 2012.

[7] K. Barr et al. The VMware mobile virtualization platform: is that a hypervisor in your pocket? *ACM SIGOPS Operating Systems Review*, 44(4):124–135, 2010.

[8] M. Satyanarayanan et al. The Case for VM-Based Cloudlets in Mobile Computing. *IEEE Pervasive*, 8(4):14–23, 2009.

[9] M.A. Hassan et al. An investigation of different computing sources for mobile application outsourcing on the road. In *Proc. of the Intl. Conf. on MOBILe Wireless MiddleWARE, Operating Systems, and Applications (Mobilware)*, 2011.

[10] T. Verbelen et al. Dynamic deployment and quality adaptation for mobile augmented reality applications. *Journal of Systems and Software*, 84(11), 2011.